LAW OFFICES
ANTONELLI, TERRY, STOUT & KRAUS, LLP
SUITE 1800
1300 NORTH SEVENTEENTH STREET
ARLINGTON, VIRGINIA 22209

DONALD R. ANTONELLI
DAVID T. TERRY
MELVIN KRAUS
WILLIAM I. SOLOMON*
GREGORY E. MONTONE
RONALD J. SHORE
DONALD E. STOUT
ALAN E. SCHIAVELLI
JAMES N. DRESSER
CARL I. BRUNDIDGE*
PAUL J. SKWIERAWSKI*

RANDALL S. SVIHLA
DAVID S. LEE*
ROBERT M. BAUER
DEMETRA J. MILLS
HUNG H. BUI*

*ADMITTED OTHER THAN VA

OF COUNSEL
CHITTARANJAN N. NIRMEL, PHD*

PATENT AGENT
LARRY N. ANAGNOS

TELEPHONE
(703) 312-6600
FACSIMILE
(703) 312-6666
E-MAIL
email@antonelli.com

May 24, 1999

Honorable Commissioner of
Patents and Trademarks
Washington, D.C. 20231

Attorney Docket Number: 500.37238X00

Sir:

Attached please find the application papers of Susumu KOBAYASHI, Youichi YAMAMOTO, Yoshito KAMEGI, covering new and useful improvements in METHOD AND SYSTEM FOR MANAGING DATABASE HAVING A CAPABILITY OF PASSING DATA, AND MEDIUM RELEVANT THERETO, comprising:

Specification, Eight (8) Claims and Abstract of the Disclosure (39 pages)

English language, Combined Declaration and Power of Attorney (2 pages)

Fourteen (14) Sheets of Drawings Showing Figures 1-16

Assignment and Recording of Assignment Letter

U.S. Government Filing Fee of $760.00

U.S. Government Recording Fee of $40.00

Change of Correspondence Address

Information Disclosure Sheet Under 37 CFR 1.56 With Attached References

Please charge any shortages in the fees or credit any overpayments thereof the deposit account of Antonelli, Terry, Stout & Kraus, LLP Account No. 01-2135 (500.37238X00).

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/nac
Attachments

Please type a plus sign (+) inside this box → [+]

PTO/SB/122 (11-96)
Approved for use through 6/30/99. OMB 0651-0035
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

CHANGE OF CORRESPONDENCE ADDRESS
Application

Address to:
Assistant Commissioner for Patents
Washington, D.C. 20231

Application Number	
Filing Date	May 24, 1999
First Named Inventor	Susumu KOBAYASHI
Group Art Unit	
Examiner Name	
Attorney Docket Number	500.3723B X00

Please change the Correspondence Address for the above-identified application to:

[+] Customer Number [020457] → *Type Customer Number here*

020457
PATENT 006111 MARK OFFICE

OR

☐ Firm *or* Individual Name					
Address					
Address					
City		State		ZIP	
Country					
Telephone		Fax			

This form cannot be used to change the data associated with a Customer Number. To change the data associated with an existing Customer Number use "Request for Customer Number Data Change" (PTO/SB/124).

I am the :

☐ Applicant.

☐ Assignee of record of the entire interest.
Certificate under 37 CFR 3.73(b) is enclosed.

[X] Attorney or agent of record .

Typed or Printed Name	Carl I. Brundidge Registration NO. 29,621
Signature	
Date	May 24, 1999

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

METHOD AND SYSTEM FOR MANAGING DATABASE HAVING A CAPABILITY OF PASSING DATA, AND MEDIUM RELEVANT THERETO

BACKGROUND OF THE INVENTION

The present invention relates to a method for processing a database, a system for processing a database, and a medium relevant thereto, and more particularly to a processing method for passing data in a database system configured in a client-server manner.

Concerning a method for managing in a database a massive amount of batch data ranging to several giga bytes such as moving picture data, video data and audio data, there has been proposed a method having a BLOB (Binary Large Object) system proposed by the SQL3 ("Database Language SQL", ISO Working Draft, July 1996). When an application program treats the BLOB data, the application program uses a variable consisting of four bytes called as a locator for uniquely identifying the BLOB data and creates the variable as the BLOB data when the value is evaluated.

For example, when a database system treats a massive amount of data, it is disadvantageously necessary to secure a large memory area for a program on which a user application is running. In order to overcome this disadvantage, as a method for treating the BLOB data by holding the BLOB data in a file without using a memory

buffer for holding the massive amount of data in the application program, a file reference technique of the database managing system DB2 ("USING THE NEW DB2 IBM's Object-Relational Database System", Don Chamberlin, Morgan Kaufmann Publishers, Inc., 1996) may be used.

On the other hand, a relational database may employ a parallel database that enables to process a massive amount of data at fast speed. The parallel database may employ a function of selecting the most approximate parallel processing according to the content of query and the state of data in the parallel database (the function of which is disclosed in JP-A-6-214843).

Further, the object relational database ("OBJECT RELATIONAL DBMSs", Michael Stonebraker, translated by K. OHTA, International Thomas Publishing Japan, August 1996) has a technique of implementing a routine function of SQL3 by using the executable codes created by describing a routine function of SQL3 in a general-purpose programming language and manipulating multimedia data such as pictures through the use of the SQL using the routine.

As a method for sharing data between processes, a technique of a memory mapped file in an operating system UNIX and so forth may be referred (X/Open Portability Guide, XPG4V2).

SUMMARY OF THE INVENTION

The foregoing prior arts have the following problems.

(1) The foregoing method of the file reference is arranged to locate a file on a client node (which corresponds to a computer machine, that is, a unit composing a computer having a central processing unit as its basis) in which an application program (AP) is running, and holds BLOB data on the file. (The file treated outside of the database server is called as "an external file" hereinafter.) If the AP issues a request of outputting the BLOB data held in the server to the file, the following process is executed.

(a) The database server reads the BLOB data at the server node.

(b) The server transfers the data to a client.

(c) The client writes the data in the external file.

In this case, the transfer of massive data between the client and the server in network communications or inter-process communications disadvantageously needs a longer time.

(2) The method of the file reference is arranged so that the database server outputs the BLOB data to the external file specified by the AP.

For example, hence, for outputting plural BLOB data units obtained as a result of the database query to the corresponding external files, the following complicated process has to be carried out.

(a) The process is specified to overwrite the BLOB data when it is outputted to the external file. The

process is executed to change a title of the external file into another title or to copy the content of the external file into another file, each time one result is outputted.

(b) The process is specified to add the BLOB data when it is outputted to the external file. Then, the process is executed to divide the BLOB data from the external file and to output the divided BLOB data units to the corresponding files. If the size of the BLOB data is not clear, each time one result is outputted, the process is executed to hold the size of the added BLOB data and then divide the BLOB data as referring to the size.

Hence, for doing such a process, the description of the source code in the AP is made disadvantageously complicated.

Further, in the case of copying long BLOD data, disadvantageously, a large amount of storage is required.

(3) In the case of applying the file-reference method in the parallel database system, though the database server makes parallel accesses to plural BLOB data units by plural parallel database processes, the BLOB data unit is written to the external file one by one in one client on which the AP is running.

Hence, the concentration of load on the client brought about as a bottleneck, which serves to lower the performance.

It is an object of the present invention to provide a technique of overcoming the foregoing disadvantages (1) to (3) and speeding up the process of passing

data from a database server to a user application in a database system.

It is a further object of the present invention to provide a technique of simplifying description of source codes used for treating data to be managed by the database in a user application.

It is a yet further object of the present invention to provide a technique of speeding up passing of plural data units between a database server and a user application in a parallel database system.

The foregoing and the other objects and the novel features of the present invention will become apparent from the description of this specification and the appended drawings.

The summaries of the representative ones of the present invention disclosed in the present application will be simply described as follows.

(1) A database processing method is arranged so that in a database system in a client-server manner for treating a massive amount of data, a database server operating in a server may output to a file the massive amount of data stored in a database requested by a user application operating in a client, and the user application may obtain the massive amount of data by referring to the file to which the massive amount of data is outputted.

(2) A database processing method is arranged so that the database server may create a file identifying information used for identifying the file to which the massive

data is to be outputted, and notify the user application of the file identifying information from the database server, and the user application may obtain the massive amount of data by referring to the file with the file identifying information.

(3) A database processing method is arranged so that the user application may request the execution of a function defined in a database, the database server may execute the function according to the request given from the user application, the function may create the file identifying information of the file to which the massive amount of data is to be outputted, the function may output the massive amount of data to the file, and the function may notify the database server of the file identifying information.

(4) A database processing method is arranged so that in a parallel database arrangement plural processes for doing database processes in parallel are executed to output to a file the massive amount of data in parallel.

(5) A database processing method is arranged so that the user application may obtain the massive amount of data by referring to the file to which the massive amount of data is outputted by the database server at a node where the database server is operating.

(6) In a database processing system arranged in a client-server manner for treating a massive amount of data, the system comprises: means for enabling a database server operating in a server to output to a file a massive amount

of data stored in a database requested by a user application operating in a client; and means for enabling the user application to obtain the massive amount of data by referring to the file to which said means is executed to output the massive amount of data.

(7) A database processing system comprises: means for enabling the database server to create a file identifying information for identifying the file to which the massive amount of data is to be outputted; means for notifying the user application of the file identifying information from the database server; and means for enabling the user application to obtain the massive amount of data by referring to the file with the file identifying information obtained by the notification.

(8) A database processing system comprises: means for enabling the user application to request the execution of a function defined in the database; means for enabling the database server to execute the function according to the request given from the user application; means for creating a file identifying information of a file to which the function is executed to output the massive amount of data; and means for enabling the function to notify the database server of the file identifying information.

(9) A database processing system comprises means for enabling the plural processes for concurrently processing databases in a parallel database arrangement to output the massive amount of data to the file in parallel.

(10) A database processing system comprises: means for

enabling the user application to obtain the massive amount of data by referring to the file, to which the database server is executed to output the massive amount of data, at the same one as the node where the database server is operating.

(11) A computer-readable storage medium having recorded a program and data in a database processing system in a client-server manner, comprises: a first process in which a database server operating in the server outputs to a file a massive amount of data stored in a database requested by a user application operating in the client; and a second process of enabling the user application to obtain the massive amount of data by referring to the file to which the massive amount of data is outputted by the first process.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a conceptual view showing a summary of a process of treating a massive amount of data through an external file according to a first embodiment of the present invention;

Fig. 2 is a schematic diagram showing a hardware arrangement according to the embodiment shown in Fig. 1;

Fig. 3 is a schematic diagram showing a function arrangement of a database system according to the embodiment shown in Fig. 1;

Fig. 4 is a view showing an SQL definition sentence for defining table "employee";

Fig. 5 is a table showing row data stored in table "employee";

Fig. 6 is a view showing an SQL definition sentence for defining function "fileout()";

Fig. 7 is a flowchart showing a process executed in an application;

Fig. 8 is a view showing part of a description of a source code of the application;

Fig. 9 is a view showing a table of retrieved results in the application;

Fig. 10 is a view showing a list of retrieved results created by the application;

Fig. 11 is a flowchart showing a process ranging from a query request in an AP (Application Program) to the obtained result;

Fig. 12 is a flowchart showing a database process in an FES of a database server;

Fig. 13 is a flowchart showing a database process in a BES of a database server;

Fig. 14 is a flowchart showing a process of "fileout" mounting function;

Fig. 15 is a conceptual view showing a summary of a database system composed of one node; and

Fig. 16 is a conceptual view showing a summary of a database system having a memory mapped file.

DETAILED DESCRIPTION OF THE EMBODIMENTS

Hereafter, an embodiment of the present invention

will be described in detail with reference to the appended drawings.

Fig. 1 is a conceptual view showing a summary of a process of enabling a user application to obtain data from a database in a parallel database system according to an embodiment of the present invention.

As shown in Fig. 1, a database system 101 of this embodiment includes a node 111 served as a client in which an application program (AP) 112 is running, a node 121 in which a front-end server (FES) 122 is running, the FES 122 corresponding to a part of a function of a database server 102 which receives a query request from the AP 112, a node 131 in which a dictionary server (DS) 132 is running, the DC 132 corresponding to a part of a function of the database server 102 which manages dictionary information 133 containing a definition information and a storage structure information in the database system 101, a group of nodes 141-1, 141-2, ..., 141-n in which plural backend servers (BES) 142-1, 142-2, ..., 142-n are running, those backend servers corresponding to a part of the function of the database server 101 for executing database processes in the database system 101 in parallel, and a group of nodes 151-1, 151-2, ... 151-n for holding external files 152-1, 152-2, ..., 152-n. Those elements are connected with one another through a network 103.

The BESs 142-1, 142-2, ..., 142-n hold data 145-1, 145-2, ..., 145-n to be manipulated by the AP 112 in data stores 144-1, 144-2, ..., 144-n.

The nodes 151-1, 151-2, ..., 151-n hold the external files 152-1, 152-2, ..., 152-n in a common file system area to be accessed by both of the AP 112 and the database server 102.

The description will be oriented to the process of rapidly and simply obtaining the data stored in the database server through the effect of the AP in the system.

The AP 112 operates to transmit a query request to the FES 122 of the database server 102. This query request includes an execution request 113 of a function (corresponding to a sort of a routine of SQL3) for outputting data 145-1, 145-2, ..., 145-n to the external files 152-1, 152-2, ..., 152-n.

The FES 122 analyzes the query request from the AP 112. In analyzing the query request, the process is executed to obtain the storage structure information and information about the execution of the function in the data stores 144-1, 144-2, ..., 144-n by referring to the dictionary information 133 of the DS 132 (arrow 162). Next, the FES 122 creates an execution procedure code 123 for the database process based on the analyzed information of the query request, sends out the code 123 to the BESs 142-1, 142-2, ..., 142-n, and requests the BESs to do the database process (arrows 163-1, 163-2, ..., 162-n). The execution procedure code 123 contains an execution indication of a function having a capability of writing the data 145-1, 145-2, ..., 145-n to the external files 152-1, 152-2, ..., 152-n.

Please charge any shortages in the fees or credit any overpayments thereof the deposit account of Antonelli, Terry, Stout & Kraus, LLP Account No. 01-2135 (500.37238X00).

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/nac
Attachments

The BESs 142-1, 142-2, ..., 142-n read the data 145-1, 145-2, ..., 145-n from the data stores 144-1, 144-2, ..., 144-n according to the execution procedure code 123, executes function mounting codes 143-1, 143-2, ..., 143-n of the function of writing the data out to the external files, and create the external files 152-1, 152-2, ..., 152-n (arrows 164-1, 164-2, ..., 164-n).

As a result of executing the function mounting codes 143-1, 143-2, ..., 143-n, the BESs 142-1, 142-2, ..., 142-n receive the titles of the external files as the executed results. After creating the external files, the BESs 142-1, 142-2, ..., 142-n return back the executed results to the FES 122.

The FES 122 edits the executed results from the BESs 142-1, 142-2, ..., 142-n to return them back to the AP 112 as the processed results, (arrows 165-1, 165-2, ..., 165-n).

The AP 112 obtains the external file names (166-1, 166-2, ..., 166-n) contained in the returned processed results to access the nodes having the external files 152-1, 152-2, ..., 152-n, (arrows 167-1, 167-2, ..., 167-n) and refer to the data held in the external files (arrows 168-1, 168-2, ..., 168-n).

The foregoing process allows the direct transfer of the data between the client and the server to be eliminated, thereby being able to speed up the process.

Moreover, since each file name of the external files is created by the database server, it is possible to

easily describe the source code in the user application.

Further, since the database server allows the plural external files to be written out in parallel, the process of outputting the external files may be sped up.

Fig. 2 is a schematic diagram showing a hardware arrangement of the embodiment shown in Fig. 1.

As shown in Fig. 2, in the database system 101, nodes 111, 121, 131, 141-1, 141-2, ..., 141-n, 151-1, 151-2, ..., 151-n each of which corresponds to the processing unit are connected through a network 103 such as a LAN (Local Area Network), so that those nodes may do communication with each other through the network 103.

Each of the nodes 111, 121, 131, 141-1, 141-2, ..., 141-n, 151-1, 151-2, 151-n includes a general computer arrangement, and comprises a data processing device 201a, 201b, 201c, 201d or 201e, a data I/O device 202a, 202b, 202c, 202d or 202e such as a keyboard, a mouse and a display, and a data storage device 203a, 203b, 203c, 203d or 203e such as a disk device. The data processing device 201a, 201b, 201c, 201d or 201e is composed of a central processing unit (CPU) 211a, 211b, 211c, 211d or 211e, an operating system (OS) 212a, 212b, 212c, 212d or 212e, a communication controller 213a, 213b, 213c, 213d or 213e connected to the network 103, a main storage unit (memory) 214a, 214b, 214c, 214d or 214e, an I/O controller 215a, 215b, 215c, 215d or 215e, and a system bus 216a, 216b, 216c, 216d or 216e for connecting those components. The data I/O device 202a, 202b, 202c, 202d or 202e is

connected to the I/O controller 215a, 215b, 215c, 215d or 215e.

The function of the program shown in Figs 1 and 3 is implemented by executing the programs stored in the memories 214a, 214b, 214c, 214d and 214e. The execution of the programs are done by the CPUs 211a, 211b, 211c, 211d and 211e under the control of the OSs 212a, 212b, 212c, 212d and 212e.

Fig. 3 is a schematic diagram showing a function arrangement of the database system in the embodiment of the present invention shown in Fig. 1.

The database system 101 comprises a node 111 of a client in which the AP 112 is running, a node 121 in which the FES 122 for receiving a query request from the AP 112 is running, a node 131 in which a dictionary server (DS) 132 for managing the dictionary information 133 is running, a plurality of nodes 141-1, 141-2, ..., 141-n in which the BESs 142-1, 142-2, ..., 142-n for executing the database process in parallel are running, and nodes 151-1, 151-2, ..., 151-n for holding the external files 152-1, 152-2, ..., 152-n.

Those nodes are mutually connected through the network 103. The AP 112 comprises an application portion 114 containing a code for doing a data process in response to a user's request, and a client communication portion 115 for managing the communication with a database server. The application portion 114 includes a code 113 for requesting the execution of a function of outputting the data to the

external file.

The FES 122 comprises an FES communication portion 124 for managing a reception of a request from the client and a reply of the database processed result to the client, and a request analyzing portion 125 for analyzing the request from the client to generate the execution procedure code 123 indicating the executing procedure of the database process.

Each of the BESs 142-1, 142-2, ..., 142-n comprises a BES communication portion 146 for managing a reception of a request from the FES 122 and a reply of the database processed result to the FES 122, a process execution portion 147 for doing a database process according to an indication contained in the execution procedure code 123, and each of data access managing portions 148-1, 148-2, ..., 148-n for managing access onto the data stores 144-1, 144-2, ..., 144-n holding the data.

Later, the description will be oriented to the database process, to which the present invention is applied, with an example of an application in detail.

The following description will be expanded with an example of the management of information about employees in the database system 101. The AP 112 is assumed to have a function of querying the names and the photos of all the employees belonging to the design division of the company and displaying the results in a list.

The information about employees is represented by table "employee". Table "employee" includes column "empno"

for indicating the employee's numbers, column "name" for indicating the employee's names, column "dept" for indicating the divisions to which the employees belong, and column "photo" for indicating the photos of the employees.

Fig. 4 shows the SQL definition sentence 401 for defining table "employee". Hereafter, the meaning of each row will be described.

402: Defines the table titled as "employee", which consists of the following columns.

403: INTEGER type column, the title of which is "empno".

404: VARCHAR type (within 30 bytes) column, the title of which is "name".

405: VARCHAR type (within 30 bytes) column, the title of which is "dept".

406: BLOB type (within 10 mega bytes) column, the title of which is "photo".

By doing a general database process on the definition sentence 401, the definition of the table is registered in the dictionary information 133.

Based on the registered definition, the database server 102 will refer to the dictionary information 133 to obtain the information about the column composition in the table and the information required for accessing the data stored according to the definitions in the table.

Fig. 5 is a schematic view showing the table in which the data about the employees is stored based on the definition 401.

The table 501 for storing the row data consists of column "empno" 511, column "name" 512, column "dept" 513, and column "photo" 514.

The meaning of the row data is indicated as follows.

521: The values of columns "empno", "name" and "dept" are INTEGER type numeric value "1789", VARCHAR type character string "George", and VARCHAR type character string "design". The value of column "photo" is an identifying information ("blob1") of the BLOB data.

This data 521 represents that the employee's number is "1789", the employee's name is "George", the belonging division of the employee is the design division, and the photo data is the BLOB data identified by "blob1".

In addition, the column values of the BLOB data are stored in such a common manner that the identifying information of the BLOB data is held in the column value and the entity of the BLOB data is held in another area of the data store.

Later, the similar row data will be briefly described.

522: The values of columns "empno", "name" and "dept" are INTEGER type numeric value "1789", VARCHAR type character string "John", and VARCHAR type character string "account". The value of column "photo" is an identifying information ("blob2") of the BLOB data.

523: The values of columns "empno", "name" and "dept" are INTEGER type numeric value "1801", VARCHAR type

character string "Thomas", and VARCHAR type character string "design". The value of column "photo" is an identifying information ("blob3") of the BLOB data.

524: The values of columns "empno", "name" and "dept" are INTEGER type numeric value "1809", VARCHAR type character string "James", and VARCHAR type character string "general". The value of column "photo" is an identifying information ("blob4") of the BLOB data.

525: The values of columns "empno", "name" and "dept" are INTEGER type numeric value "1829", VARCHAR type character string "Andrew", and VARCHAR type character string "account". The value of column "photo" is an identifying information ("blob5") of the BLOB data.

526: The values of columns "empno", "name" and "dept" are INTEGER type numeric value "1837", VARCHAR type character string "Martin", and VARCHAR type character string "planning". The value of column "photo" is an identifying information ("blob6") of the BLOB data.

In turn, the description will be oriented to function "fileout()" for providing a function of writing the BLOB data to the external file.

The interface of function "fileout()" is specified so that an input (argument of the function) is the BLOB type data and an output (return value of the function) is the VARCHAR type (within 255 bytes) character string.

This function "fileout()" has a capability of writing the content of the BLOB data inputted as the

argument to the external file. The name of the external file to which the content of the BLOB data is to be written is created by this function itself. Then, the name of the external file to which the content of the BLOB data is to be written is returned as the return value of the function.

At the nodes 151-1, 151-2, ..., 151-n shown in Fig. 3, the external files are created in the common file system area.

The processing content of this function "fileout()" will be described with reference to Fig. 14.

The function "fileout()" is defined by the SQL definition sentence as shown in Fig. 6.

The meaning of each row of the definition sentence 601 of the function will be indicated as below.

602: Defines the function named as "fileout", the argument of which is a BLOB type value.

603: The return value is a VARCHAR type (within 255 bytes) value.

604: The executable code mounting this function (function mounting code) is held in a file named "bin/fileout".

605: The source code of the function mounting code is described in the C language.

By doing a general database process based on the definition sentence 601, the definition of the function is registered in the dictionary information 133 (see Fig. 3).

Based on the registered definition, the database server 102 will refer to the dictionary information 133 to

obtain the information required for executing the function, such as an I/O of the function "fileout()" and the name of the function mounting code "bin/fileout".

Fig. 7 is a flowchart showing a summary of a process executed in the application portion 114 of the AP 112.

This flowchart shows a process (701) of displaying the photo data of all the employees belonging to the design division in a list.

At first, the process is executed to issue a query request of "obtain the names and the photo data of all the employees belonging to the design division" (step 702).

Next, the process is executed to pick up the result of the query request (step 703). It is determined if data can be obtained in the picked result (step 704). If data exists in the picked result, the result data is set to a list (step 705). Then, the process returns to the step 703.

If no data exists in the result picked at the step 704, that is, if data in all the retrieved results is obtained, the list of the result data (containing the pictures of the photo data) is displayed (step 706), and then the process is terminated (step 707).

Fig. 8 shows a part of a source code for creating the application for doing the process shown in Fig. 7.

The meaning of each row of a part 801 of the source code included in the application is indicated below.

802: Start the declaration of an SQL variable.

803: Declare variable "photoFilename" (hold a name of an external file to which the photo data is to be outputted).

804: Declare variable "empName" (hold the names of the employees).

805: Finish the declaration of the SQL variable.

806: Define cursor "curs1" for an SQL retrieval sentence of the rows 807 to 809.

807: The retrieval projection item is a call of function "fileout()" with columns "name" and "photo" as the arguments.

808: The table to be retrieved is "employee".

809: The retrieval condition is that the value of column "dept" is character string value "design" (which means that the belonging division is the design one).

810: Request to open a cursor.

811: The process up to the row 815 is repeated.

812: Request to fetch cursor "curs1". The result is held in variables "empName" and "photoFilename".

813: If no data exists in the fetched result, the process goes out of the repetitive process.

814: Call function "setImageDataToList()". The manes of the employees and the external file name are passed to the arguments. Herein, function "setImageDataToList()" is a code linked to the application though not shown. It has a function of setting the resulting data to the data having a general list structure.

815: Indicate the end of the repetitive process range from the row 811.

816: Request to close the cursor.

817: Call function "displayImageDataList()".

Herein, function "displayImageDataList ()" is a code linked to the application though not shown. This function has a simple data manipulating capability of creating a HTML (Hypertext Markup Language) text file of the result list based on the list of the resulting data set in function "setImageDataToList()" and displaying the list through the use of the HTML browser. In displaying the list, the pictures of the photo data are also displayed in a thumbnail manner after reading the picture file.

The source code 801 utilizes table "employee" defined in Fig. 4 and function "fileout()" shown in Fig. 6.

The database server has already returned back the name of the external file. This thus eliminates the necessity of creating the file name and copying the file in the source code. This serves to simplify the description of the AP.

Further, using the source code 801, the AP 112 is created in the following general procedure.

(1) The source code of the application is converted into the source code of the host language through a preprocessor contained in a tool for developing the application generally belonging to the database system.

(2) Using the compiler of the host language, the converted source code is converted into the object code.

This object code is made to be the application portion 114 of the AP 112. In particular, the function execution request 113 is created based on the portion for calling the function with the SQL sentence (rows 806 to 810 and 811).

(3) The AP is created by linking to the library containing the code having the function of the client communication portion 115.

Fig. 9 shows the retrieved result obtained by the process shown in Figs. 11 to 14 during the process of the AP 112 shown in Figs. 7 and 8.

The result of retrieving the database is made to be a table 901. The table 901 contains a row having a combination of a name of an employee belonging to the design division and a name of the external file holding the picture data about the employee.

The table 901 of the retrieved result consists of a column (911) for "name" specified to a projection item of a retrieval sentence and a column (912) for "fileout-(photo)" specified to a projection item of the retrieval sentence.

The meaning of each row data is indicated as follows.

921: The column 911 has a value of "George". The column 912 has a value of "dbsvexfile1".

It indicates that the name of the employee is George and the photo data of the employee is held in the external file named "dbsvexfile1".

922: The column 911 has a value of "Thomas". The

column 912 has a value of "dbvexfile2".

It indicates that the name of the employee is Thomas and the photo data of the employee is held in the external file named "dbsvexfile2".

Fig. 10 shows the HTML text of the result list created by the AP 112 based on the retrieved result shown in Fig. 9.

The meaning of each row of the HTML text (1001) is indicated as below.

1002: Start of HTML text.

1003: It indicates that character string "George" and the picture data held in file name "dbsvexfile1" are displayed, and then the sentence is line-fed.

1004: It indicates that character string "Thomas" and the picture data held in file name "dbsvexfile2" are displayed and then the sentence is line-fed.

1005: End of HTML text

The AP 112 is executed to apply this HTML text 1001 into the common HTML browser function for displaying the results in a list (concretely, the manes of the employees and the photos of the employees). The picture data in files "dbsvexfile1" and "dbsvexfile2" are read from the HTML browser function and then is displayed.

As described above, the AP enables to directly obtain the data with the external file name without having to receive the data transferred from the database server in communication.

Fig. 11 is a flowchart showing a summary of a

process ranging from a query request to the acquisition of the result in the AP 112.

It indicates a flow of the general process in the case of executing the database process such as the forgoing application.

At the step 703 of the flowchart shown in Fig. 7, the process (1101) for issuing the query request and obtaining the result is indicated.

At first, the AP 112 executes the query request contained in the application portion 114 (step 1102). This query request contains a process of calling a function of a database process executing request transmission provided by the client communication portion 115.

Next, the client communication portion 115 operates to transmit the query request to the FES 122 of the database server 102 (step 1103). After the query request is received in the FES 122, the details of the database process will be described below with reference to Figs. 12 to 14.

The client communication portion 115 enters into a waiting state for a reply from the FES 122 (step 1104). When it receives the reply from the FES 122 (step 1105), the client communication portion 115 operates to edit the result of the database process returned back thereto, to pass it to the application portion 114 (step 1106), and then to terminate the process of its own (step 1107).

Fig. 12 is a flowchart showing the database process (step 1201) executed in the FES 122 at the step

1103 in Fig. 11.

At first, the FEE communication portion 124 in the FES 122 receives the query request from the client communication portion 115 (step 1202).

Next, the query request received by the request analyzing portion 125 is analyzed (step 1203). In analyzing the query request, the request analyzing portion 125 requests the dictionary information 133 to the DS 132, and then obtains the information used for accessing the data requested by the AP and the information used for executing the function by referring to the dictionary information 133.

Herein, the retrieval request is recognized for table "employee". Then, the process is executed to select the procedure of sequentially processing the row data for meeting the condition about the row data of table "employee" stored as shown in Fig. 5 based on the definition shown in Fig. 4.

Further, in the query, it is recognized that the execution request for function "fileout()" is issued. Based on the definition shown in Fig. 6, the information used for executing the function execution code 143 is obtained.

In succession, the request analyzing portion 125 creates the execution procedure code 123 for doing the database process in the BES 142-1, 142-2, ..., 142-n based on the result analyzed in step 1203 (step 1204).

The execution procedure code 123 contains a code

about the procedure of accessing the row data stored in table "employee" as shown in Fig. 5 and a code about the procedure of executing the function mounting code 143.

Next, the FES communication portion 124 operates to transmit the process execution request to the BESs 142-1, 142-2, ..., 142-n so that the database process may be executed by using the execution procedure code 123 (step 1205).

The database process in the BESs 142-1, 142-2, ..., 142-n will be described in detail in Fig. 13.

Next, the FES communication portion 124 waits for a reply from the BESs 142-1, 142-2, ..., 142-n (step 1206).

The FES communication portion 124 determines if all the replies from the BESs 142-1, 142-2, ..., 142-n are received (step 1207). If all the replies are received, the FES communication portion 124 edits the database process result contained in the replies, and returns back the edited result to the client communication portion 115 that has issued the query request (step 1208), and then terminates the process of its own (step 1209).

Fig. 13 is a flowchart showing a summary of a process of executing the database process according to the execution procedure code 123 in the BESs 142-1, 142-2, ..., 142-n.

A process (step 1301) is executed according to the database process executing request at the step 1205 shown in Fig. 12.

At first, the BES communication portion 146 (see

Fig. 3) receives the database process executing request from the FES communication portion 124 (step 1302).

This database process executing request contains the execution procedure code 123.

Next, the process executing portion 147 analyzes the execution procedure code 123 contained in the communication data (step 1303).

Herein, the process is executed to obtain only the values of columns "name" and "photo" about the row data for meeting the condition contained in the row data of table "employee" stored as shown in Fig. 5. Then, it is recognized that the sequential function mounting code 143 is to be executed. Later, the process is proceeded along the execution procedure recognized by the analysis.

The process execution portion 147 (see Fig. 3) gives the data access managing portion 148 a request for obtaining the row data for meeting the retrieval condition according to the analyzed result of the execution procedure code 123 at the step 1303.

Herein, the process execution portion 147 requests the row data that the value of column "dept" is character string "design".

Next, it is determined if the row data can be obtained (step 1305). If it is obtained, the function mounting code 143 ("bin/fileout") is executed for the value of the obtained column "photo" according to the analyzed result of the execution procedure code 123 (step 1306).

The program for operating the BES is dynamically

linked with the function mounting code "bin/fileout" when it is in execution. At this time, the BLOB data that is the column value of column "photo" is passed as an argument to the program. For passing the data, it is possible to use the BLOB locator of the SQL3. The BLOB data may be manipulated through the locator.

As a result of executing the function mounting code, the VARCHAR type value that is the name of the external file can be obtained as a return value.

The process of the function mounting code "bin/fileout" will be described with reference to Fig. 14.

Next, the process is executed to create and hold the column data of the database process result based on the obtained column value and the return value of the function (step 1307), and then to repeat the process from the step 1304.

Further, if no data for meeting the retrieval condition can be obtained at the step 1305, that is, if all data for meeting the retrieval condition have been already obtained, the column data of the database process result is edited (step 1308), and then the process is terminated (step 1309).

This process is executed to process the row data 521 and 523 belonging to the design division, that is, column "dept" having a value of "design" (the data corresponding to the employees George and Thomas) and to create the data on which the column data 921 and 922 of the result table shown in Fig. 9 (the column data having a combination

of an employee's name and an external file name).

Fig. 14 is a flowchart showing a summary of a process of "bin/fileout" that is the mounting code of function "fileout()".

Fig. 14 shows the process (step 1401) of "bin/fileout" executed at the step 1306 shown in Fig. 13.

At first, the external files 152-1, 152-2, ..., 152-n are created and opened on the areas of the common file system to the nodes 151-1, 151-2, ..., 151-n shown in Fig. 3 (step 1402).

Herein, the name of the external file is created by using function "tempnam()" of the C library. Hence, the external file can be uniquely identified by this name.

Next, the process is executed to read the BLOB data inputted at an argument, and to write the data onto the file opened at the step 1402 (step 1403).

Next, the external file is closed (step 1404), and then the process is terminated (step 1406) with the external file name as the return value (step 1405).

Herein, the external file name of the return value is passed to the AP 112 as the processed result. The AP 112 enables to refer to the external files 152-1, 152-2, ..., 152-n that are located in the common file system area so that the AP 112 can obtain the data without having to transfer the data between the AP and the database server.

Further, the identifier name of the external file is created by the database server, so that the description

of the source code of the application is made simpler as mentioned above.

Moreover, the parallel database process is arranged so that the BESs 142-1, 142-2, ..., 142-n may output the BLOB data to the corresponding external files in parallel. It means that this parallel database process offers a faster process than the foregoing prior art arranged so that only the client outputs the data.

Next, the description will be oriented to another embodiment of the present invention with reference to Fig. 15.

Fig. 15 is a conceptual view showing a summary of a database system composed of one node. This is an example of a system for saving the process cost of the file access between the plural nodes and thereby speeding up the acquisition of the data.

The database system 101-a is composed of one node 1501, the function arrangement of which is the same as the arrangement shown in Figs. 1 and 3. Fig. 15 shows only the main functions.

In this arrangement, the same process as the process of the foregoing AP 112 is executed.

That is, the AP 112-a issues a query request to the database server 102-a (step 1511). Then, the database server 102-a executes the mounting function code 143-a, reads out the BLOB data 145-a held in the data store 144-a (step 1512), and then outputs the BLOB data 145-a to the external file 152-a (step 1513). Then, the AP 112-a

operates to refer to the external file 152-a for obtaining the BLOB data 145-a (step 1514).

In this process, the flow of the data until the BLOB data 145-a is obtained by the AP 112-a is indicated as follows.

(1) The database server 102-a reads the BLOB data 145-a.

(2) The database server 102-a outputs the BLOB data 145-a to the external file 152-a.

(3) The AP 112-a reads the BLOB data 145-a from the external file 152-a.

During this interval, no BLOB data is communicated through a network or between the processes. Hence, as compared with the foregoing prior art in which the data is inevitably transferred, this embodiment may offer a faster process speed.

In turn, the description will be oriented to another embodiment of the present invention with reference to Fig. 16.

Fig. 16 is a conceptual view showing a summary of a database system arranged to use a memory mapped file. This database system 101-b uses the memory mapped file 1602 in place of the external file. The other basic system arrangement is the same as that shown in Fig. 1 or Fig. 15. In this embodiment, at one node 1601, the AP 112-b and the database server 102-b are operated. The database server 102-b operates to set data (152-b) to an area of the memory mapped file 1602, and then to return back a position

information 166-b containing the memory mapped file identifier and the memory address as the identifier of the area to the AP 112-b (arrow 1604). The AP operates to refer to the memory area based on the position information 166-b (arrow 1604) for obtaining the data 152-b (arrow 1606).

This thus makes it possible to speed up the AP's process of obtaining the data not by using a data storage device such as a disk but by using a faster accessible memory.

The process of the foregoing flowchart can be implemented by executing the program in the data processing device. The program can be stored in a storage medium to be accessed by a computer such as a hard-disk device or a floppy disk, so that the access to the program is made possible through the network.

In the database system arranged in a client-server manner, the database server provides means for outputting data to a storage device and enabling the user application to directly refer to the area of the data for obtaining the data. Hence, no data is required to be transferred between the client and the server, so that the fast processing is made possible.

Further, the database server provides means for creating an identifier of an area on the storage device. Hence, if the application uses two or more pieces of data, it is not necessary to sequentially specify the identifier of the storage area, which therefore makes the description

of the application simpler.

In the parallel database arrangement, the database server provides means for outputting the plural pieces of data to the storage device in parallel. This thus makes it possible to output the data to pass the data to the AP at fast speed.

According to the present invention, in the case of storing a massive amount of data in the database and treating the data in the application, the communication cost and the necessary amount of memory may be greatly reduced. This is quite advantageous.

CLAIMS

1. A database processing method used in a database system arranged in a client-server manner, comprising:

a first process of enabling a database server operating at a server to output data, which is stored in a database requested by a user application operating at a client, to an area on a storage device; and

a second process of enabling said user application to refer to said area on said storage device to which said data is outputted in said first process, to obtain said data.

2. A database processing method as claimed in claim 1, further comprising:

a third process of enabling said database server to create a storage area identifying information for identifying the area on said storage device to which said data is outputted;

a fourth process of notifying said user application of said storage area identifying information from said database server; and

a fifth process of enabling said user application to refer to the area on said storage device using said storage identifying information obtained by said notification to obtain said data.

3. A database processing method as claimed in claim 1 further comprising:

a sixth process of enabling said user application to request an execution of a function defined in said

database;

a seventh process of enabling said database server to execute said function according to a request from said user application;

an eighth process of enabling said function to create a storage area identifying information of said storage device to which said data is outputted;

a ninth process of enabling said function to output said data to said storage area; and

a tenth process of enabling said function to notify said database server of said storage area identifying information.

4. A database processing method as claimed in claim 1, further comprising:

a process of enabling plural processes, which has a parallel database arrangement and executes a database process in parallel, to output said data to said storage device in parallel.

5. A database processing method as claimed in claim 1, further comprising:

a process of enabling said user application to refer to said storage device to which said data is outputted by said database server, at the same node as a node where said database server is in operation to obtain said data.

6. A database processing system used in a database system having a client-server arrangement for treating a massive amount of data, comprising:

first means for enabling a database server operating in a server to output to a file said massive amount of data stored in a database requested by a user application operating in a client; and

second means of enabling said user application to refer to said file where said massive amount of data is outputted by said first means, to obtain said massive amount of data.

7. A database processing system as claimed in claim 6, further comprising:

means of enabling said database server to create a file identifying information for identifying said file where said massive amount of data is outputted;

means of notifying said user application of said file identifying information from said database server; and

means of enabling said user application to refer to said file by using said file identifying information obtained by said notification, to obtain said massive amount of data.

8. A computer-readable storage medium recorded a program and data in a database system arranged in a client-server manner, said program and data comprising:

a first procedure of enabling a database server operating in said server to output to a file a massive amount of data stored in a database requested by a user application operating in a client; and

a second procedure of enabling said user application to refer to said file to which said massive amount of

data is outputted by said first procedure, to obtain said massive amount of data.

ABSTRACT OF THE DISCLOSURE

In a database system of a client-server arrangement, a database server operating in a server outputs data to a storage device. Here, the data stores in a database that is requested by a user application operating in a client. The user application refers to the storage device to which the data is outputted, to obtain the data. Thereby, the data managed by the database server is passed to an application at high speed in the database system.

FIG. 1

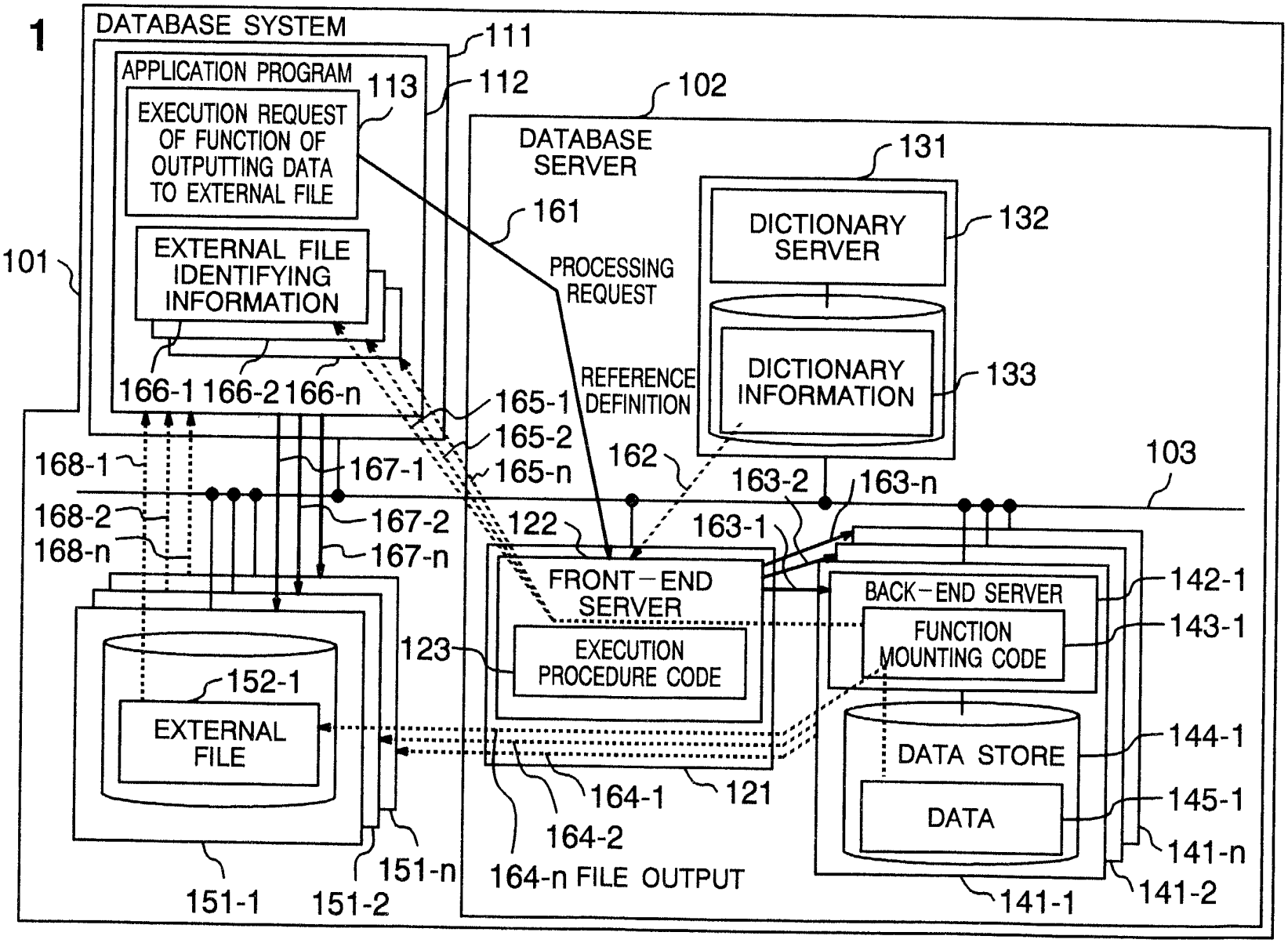

DATABASE SYSTEM
111
APPLICATION PROGRAM
113
112
102
EXECUTION REQUEST OF FUNCTION OF OUTPUTTING DATA TO EXTERNAL FILE
DATABASE SERVER
131
161
DICTIONARY SERVER
132
101
EXTERNAL FILE IDENTIFYING INFORMATION
PROCESSING REQUEST
DICTIONARY INFORMATION
133
REFERENCE DEFINITION
166-1 166-2 166-n
165-1
165-2
165-n
162
103
168-1
168-2
168-n
167-1
167-2
167-n
163-2
163-n
163-1
122
FRONT-END SERVER
BACK-END SERVER
142-1
143-1
123
EXECUTION PROCEDURE CODE
FUNCTION MOUNTING CODE
152-1
EXTERNAL FILE
DATA STORE
144-1
145-1
DATA
141-n
141-2
141-1
164-1
121
164-2
164-n FILE OUTPUT
151-n
151-1
151-2

FIG. 2

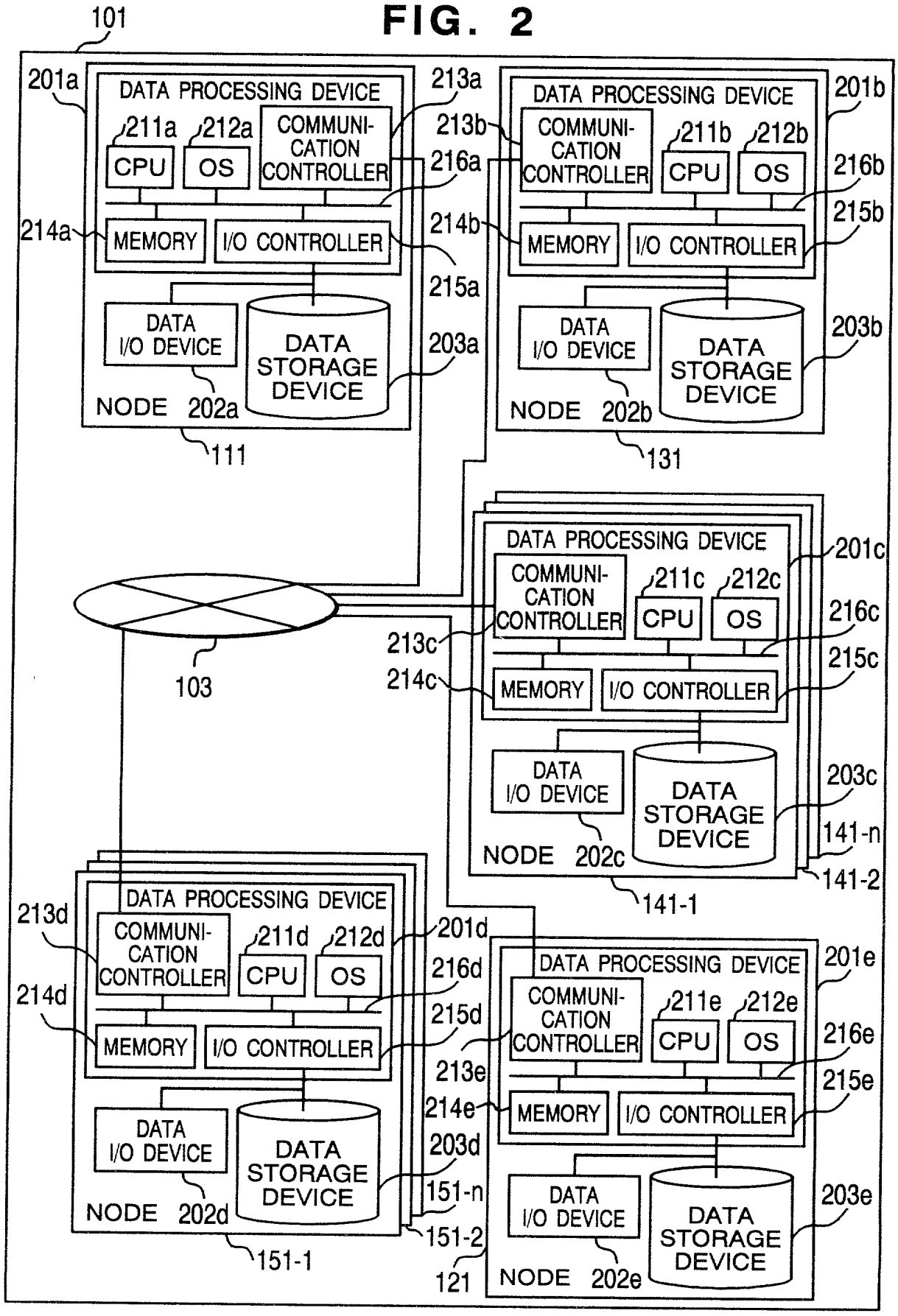

101
201a
DATA PROCESSING DEVICE
211a
212a
COMMUNI-CATION CONTROLLER
CPU
OS
213a
216a
214a
MEMORY
I/O CONTROLLER
214b
215a
DATA I/O DEVICE
DATA STORAGE DEVICE
203a
NODE 202a
111
201b
213b
211b
212b
216b
215b
203b
NODE 202b
131
103
201c
213c
211c
212c
216c
214c
215c
203c
NODE 202c
141-1
141-n
141-2
213d
201d
211d
212d
216d
214d
215d
203d
NODE 202d
151-1
151-n
151-2
121
201e
213e
211e
212e
216e
214e
215e
203e
NODE 202e

FIG. 3

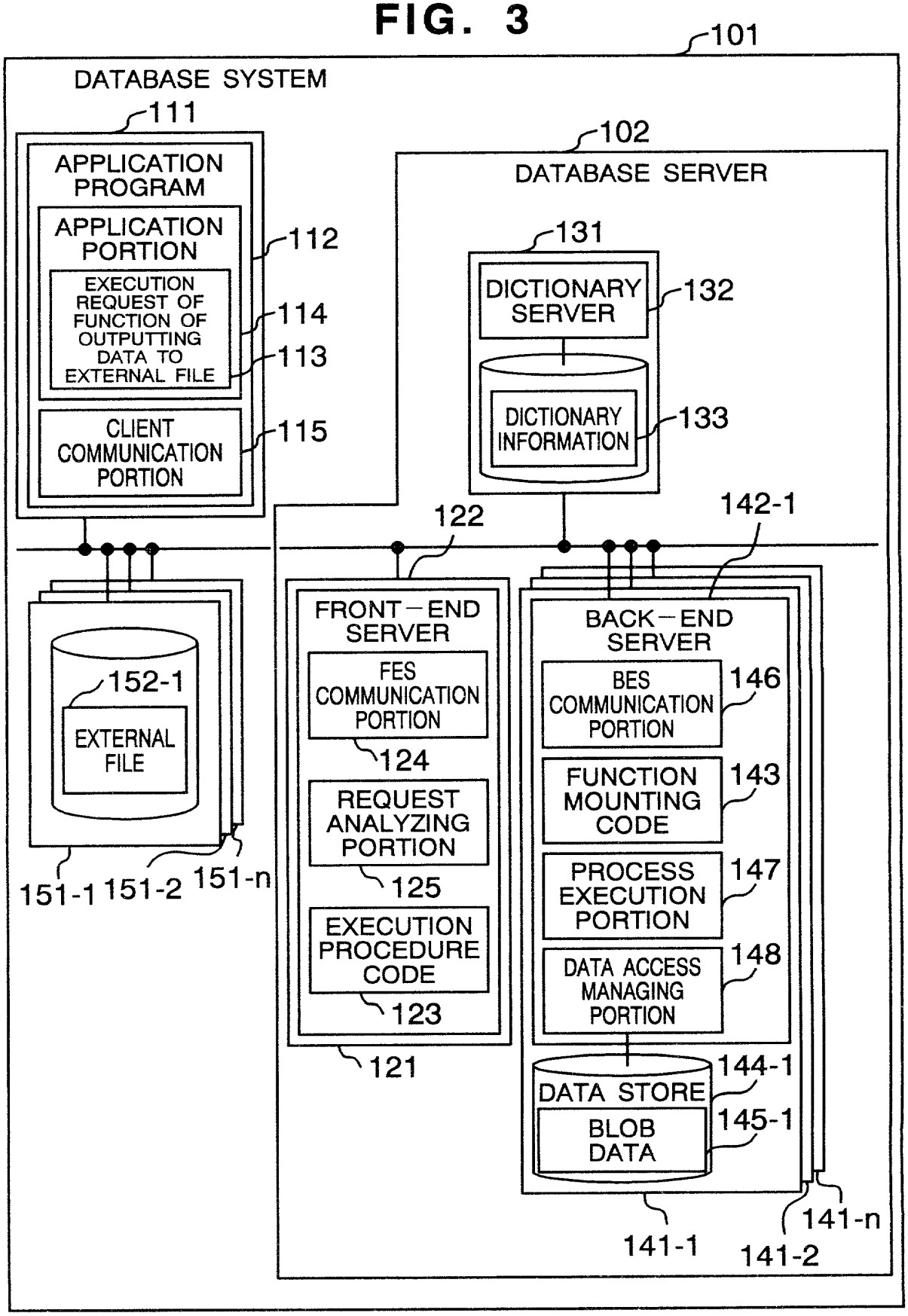

101
DATABASE SYSTEM
111
APPLICATION PROGRAM
APPLICATION PORTION
112
EXECUTION REQUEST OF FUNCTION OF OUTPUTTING DATA TO EXTERNAL FILE
114
113
CLIENT COMMUNICATION PORTION
115
102
DATABASE SERVER
131
DICTIONARY SERVER
132
DICTIONARY INFORMATION
133
122
142-1
152-1
EXTERNAL FILE
151-1
151-2
151-n
FRONT-END SERVER
FES COMMUNICATION PORTION
124
REQUEST ANALYZING PORTION
125
EXECUTION PROCEDURE CODE
123
121
BACK-END SERVER
BES COMMUNICATION PORTION
146
FUNCTION MOUNTING CODE
143
PROCESS EXECUTION PORTION
147
DATA ACCESS MANAGING PORTION
148
DATA STORE
144-1
BLOB DATA
145-1
141-1
141-2
141-n

FIG. 4

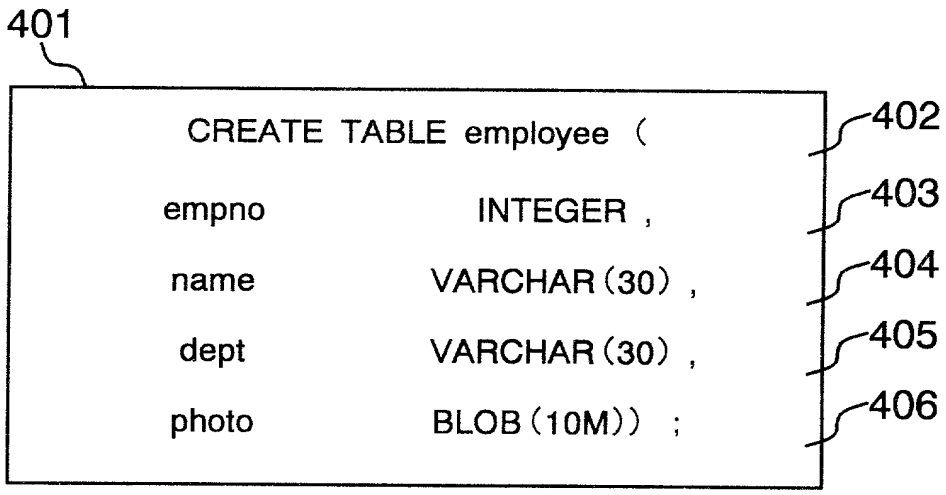


FIG. 5

501

511 empno	512 name	513 dept	514 photo
1789	George	design	(blob1)
1797	John	account	(blob2)
1801	Thomas	design	(blob3)
1809	James	general	(blob4)
1829	Andrew	account	(blob5)
1837	Martin	planning	(blob6)

521
522
523
524
525
526

FIG. 6

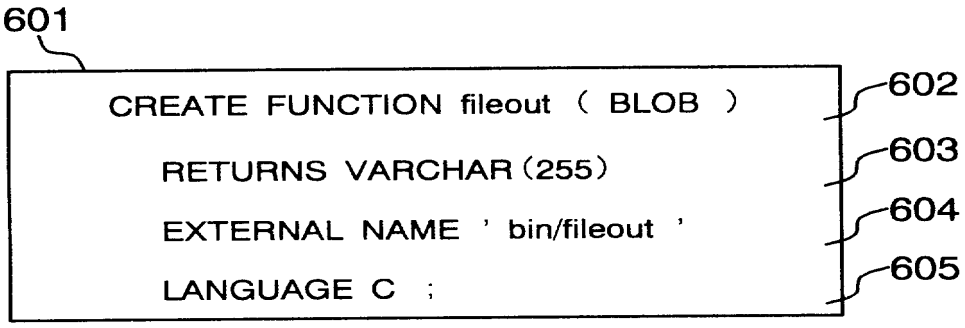

601
CREATE FUNCTION fileout (BLOB)
RETURNS VARCHAR (255)
EXTERNAL NAME ' bin/fileout '
LANGUAGE C ;
602
603
604
605

FIG. 7

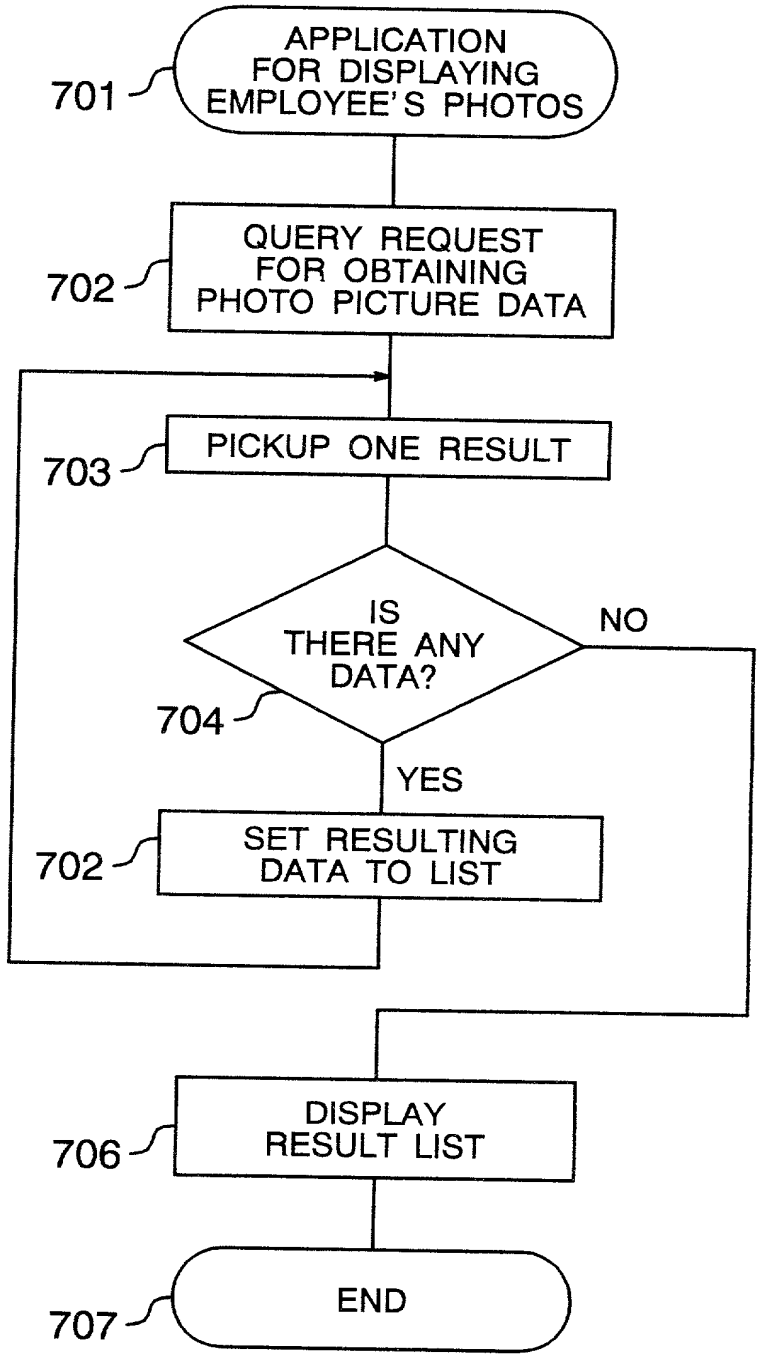

APPLICATION FOR DISPLAYING EMPLOYEE'S PHOTOS
701
QUERY REQUEST FOR OBTAINING PHOTO PICTURE DATA
702
PICKUP ONE RESULT
703
IS THERE ANY DATA?
704
NO
YES
SET RESULTING DATA TO LIST
702
DISPLAY RESULT LIST
706
END
707

FIG. 8

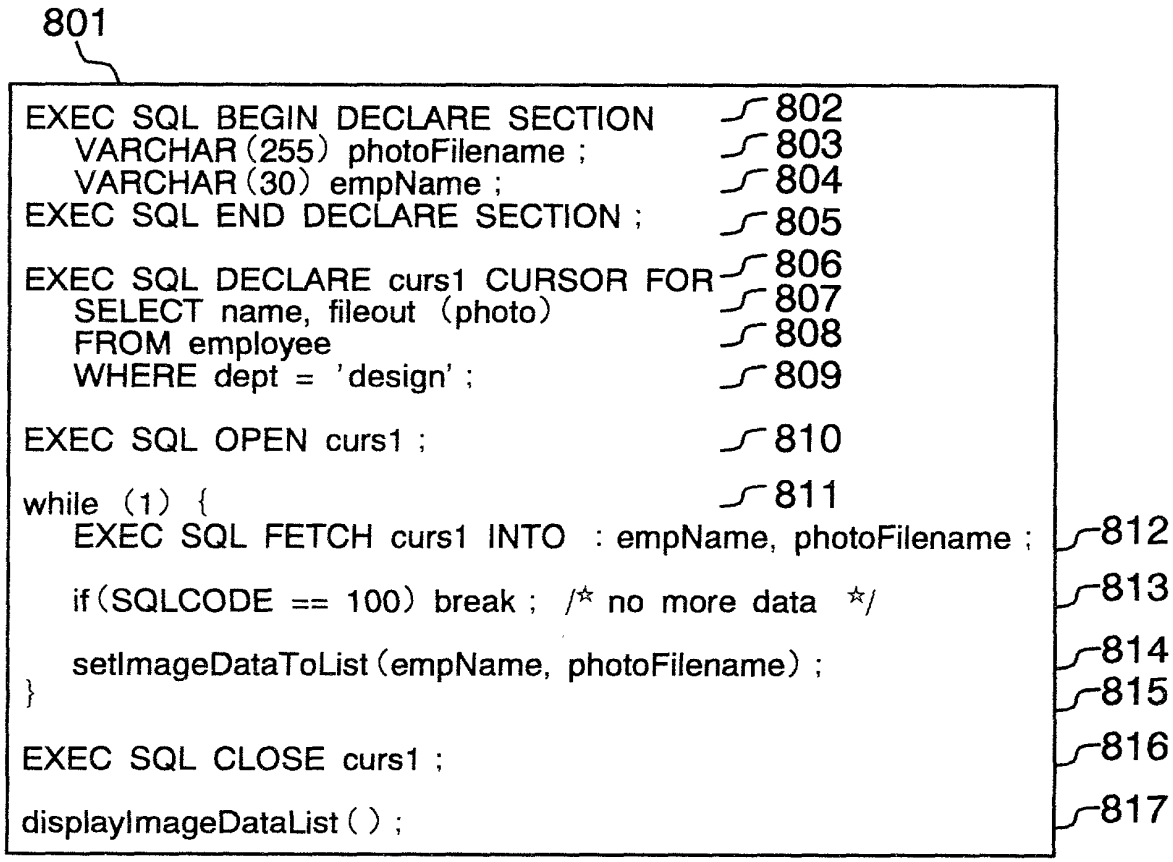

801
EXEC SQL BEGIN DECLARE SECTION
VARCHAR(255) photoFilename ;
VARCHAR(30) empName ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL DECLARE curs1 CURSOR FOR
SELECT name, fileout (photo)
FROM employee
WHERE dept = 'design' ;
EXEC SQL OPEN curs1 ;
while (1) {
EXEC SQL FETCH curs1 INTO : empName, photoFilename ;
if(SQLCODE == 100) break ; /* no more data */
setImageDataToList(empName, photoFilename) ;
}
EXEC SQL CLOSE curs1 ;
displayImageDataList ( ) ;
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817

FIG. 9

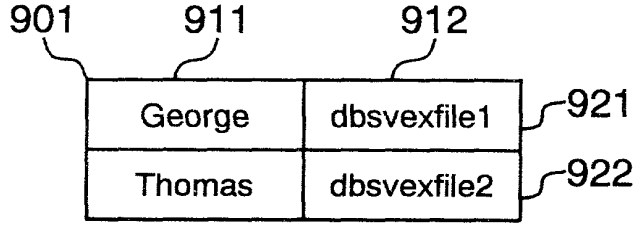

901
911
912
George
dbsvexfile1
Thomas
dbsvexfile2
921
922

FIG. 10

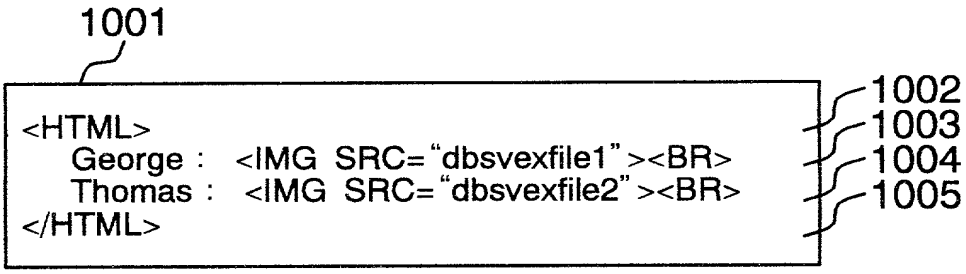

1001
<HTML>
George : <IMG SRC="dbsvexfile1">
Thomas : <IMG SRC="dbsvexfile2">
</HTML>
1002
1003
1004
1005

FIG. 11

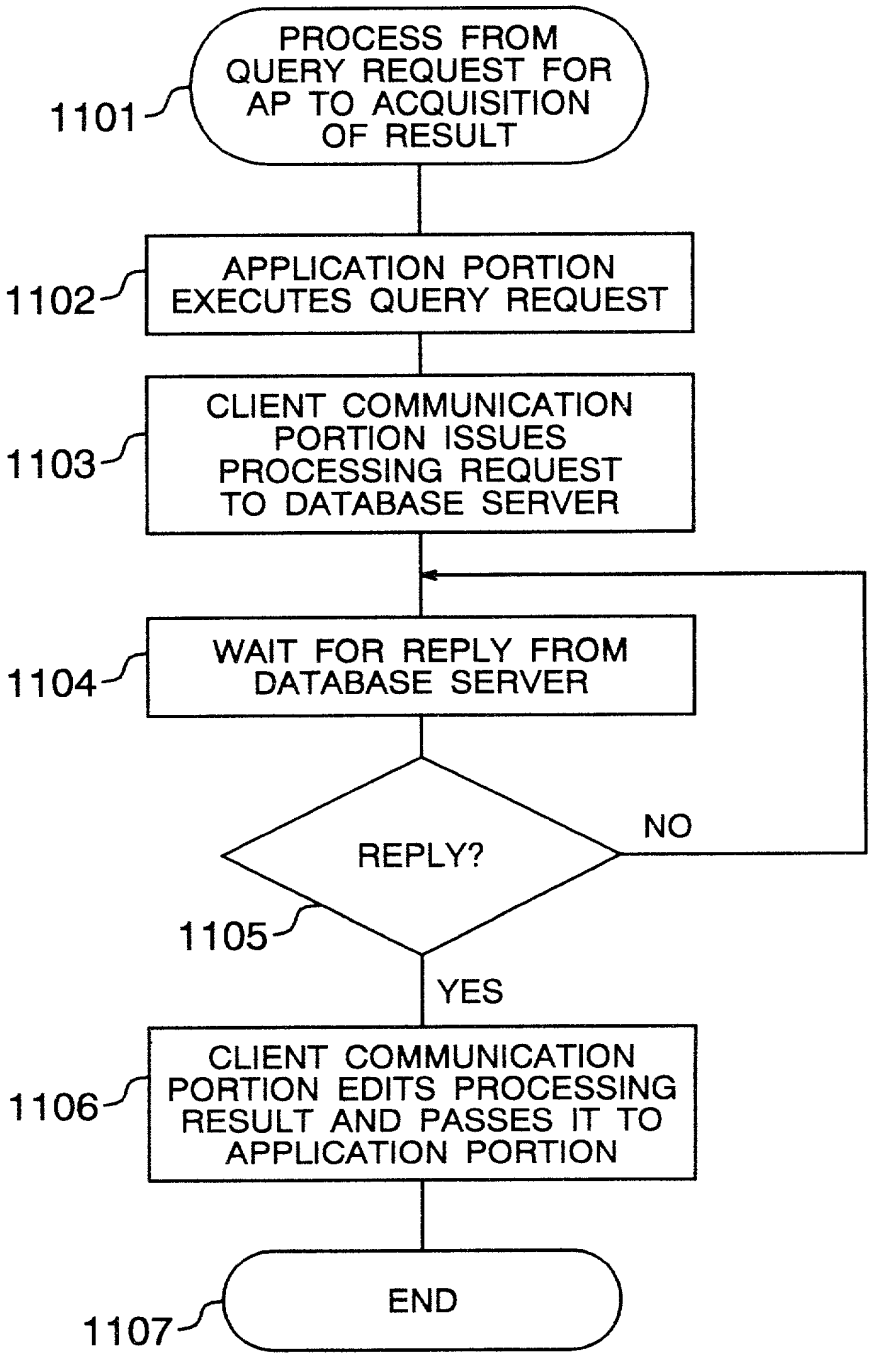

PROCESS FROM QUERY REQUEST FOR AP TO ACQUISITION OF RESULT
1101
APPLICATION PORTION EXECUTES QUERY REQUEST
1102
CLIENT COMMUNICATION PORTION ISSUES PROCESSING REQUEST TO DATABASE SERVER
1103
WAIT FOR REPLY FROM DATABASE SERVER
1104
REPLY?
NO
1105
YES
CLIENT COMMUNICATION PORTION EDITS PROCESSING RESULT AND PASSES IT TO APPLICATION PORTION
1106
END
1107

FIG. 12

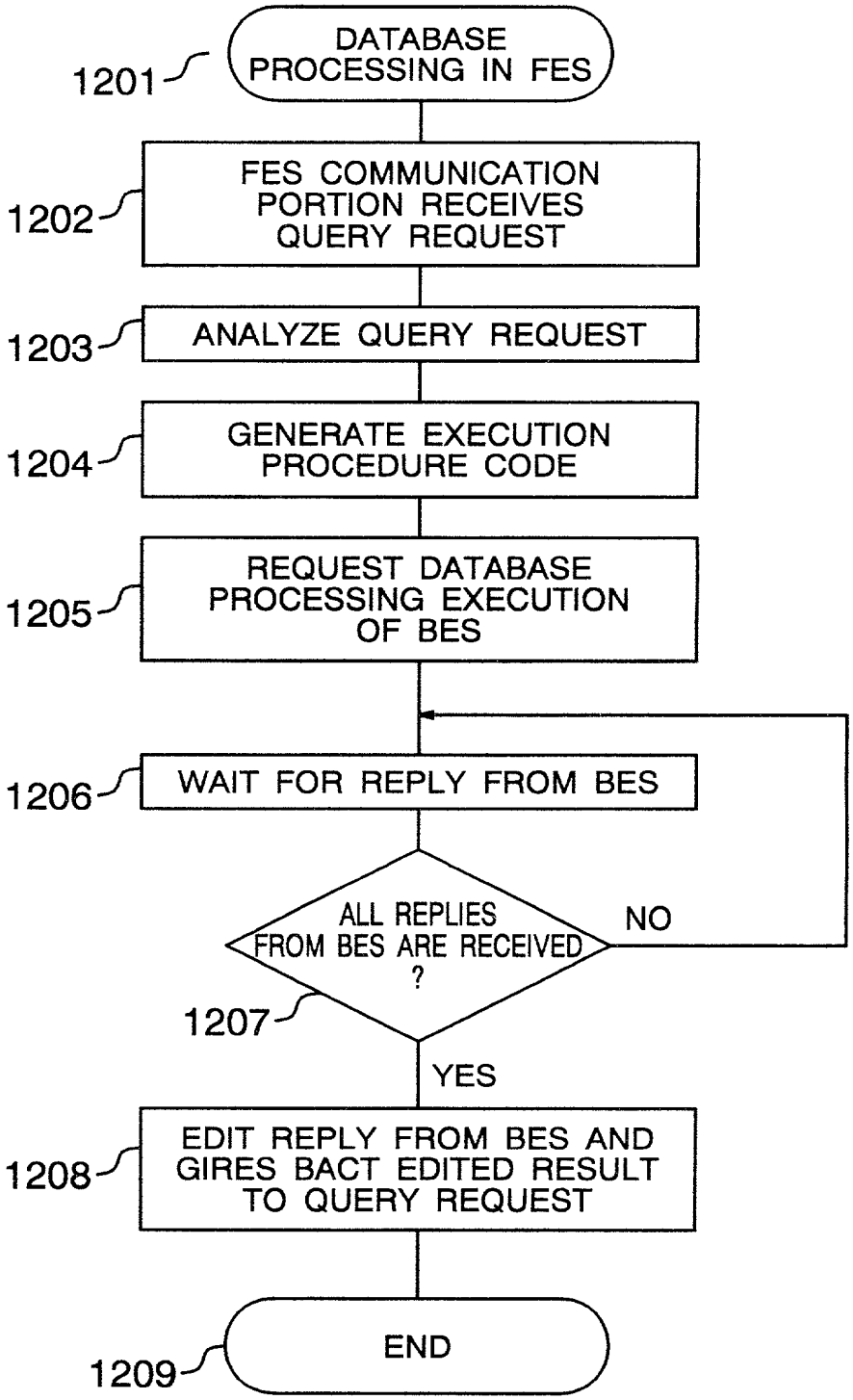

1201
DATABASE PROCESSING IN FES
1202
FES COMMUNICATION PORTION RECEIVES QUERY REQUEST
1203
ANALYZE QUERY REQUEST
1204
GENERATE EXECUTION PROCEDURE CODE
1205
REQUEST DATABASE PROCESSING EXECUTION OF BES
1206
WAIT FOR REPLY FROM BES
ALL REPLIES FROM BES ARE RECEIVED ?
NO
1207
YES
1208
EDIT REPLY FROM BES AND GIRES BACT EDITED RESULT TO QUERY REQUEST
1209
END

FIG. 13

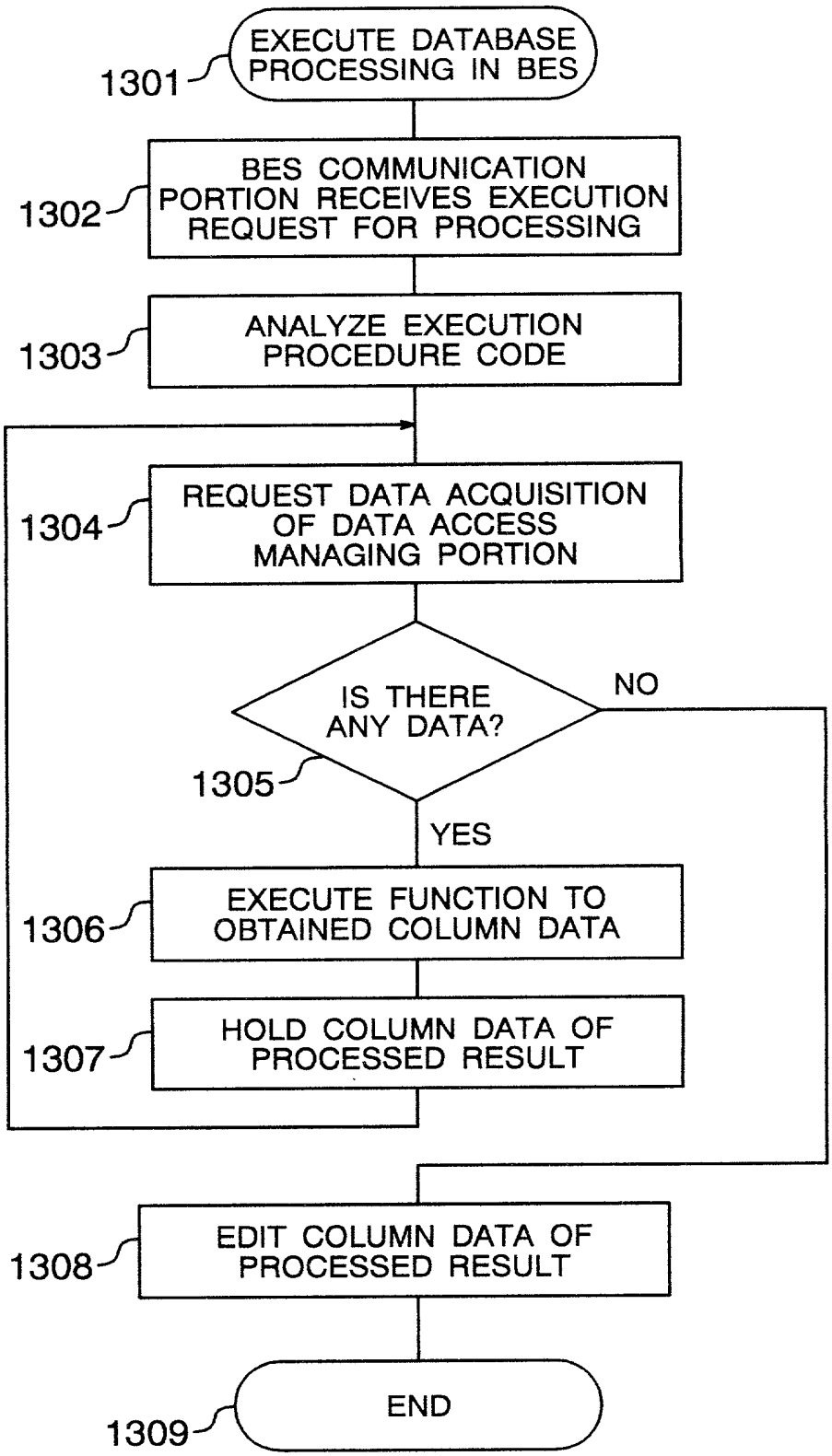

EXECUTE DATABASE PROCESSING IN BES
1301
BES COMMUNICATION PORTION RECEIVES EXECUTION REQUEST FOR PROCESSING
1302
ANALYZE EXECUTION PROCEDURE CODE
1303
REQUEST DATA ACQUISITION OF DATA ACCESS MANAGING PORTION
1304
IS THERE ANY DATA?
1305
NO
YES
EXECUTE FUNCTION TO OBTAINED COLUMN DATA
1306
HOLD COLUMN DATA OF PROCESSED RESULT
1307
EDIT COLUMN DATA OF PROCESSED RESULT
1308
END
1309

FIG. 14

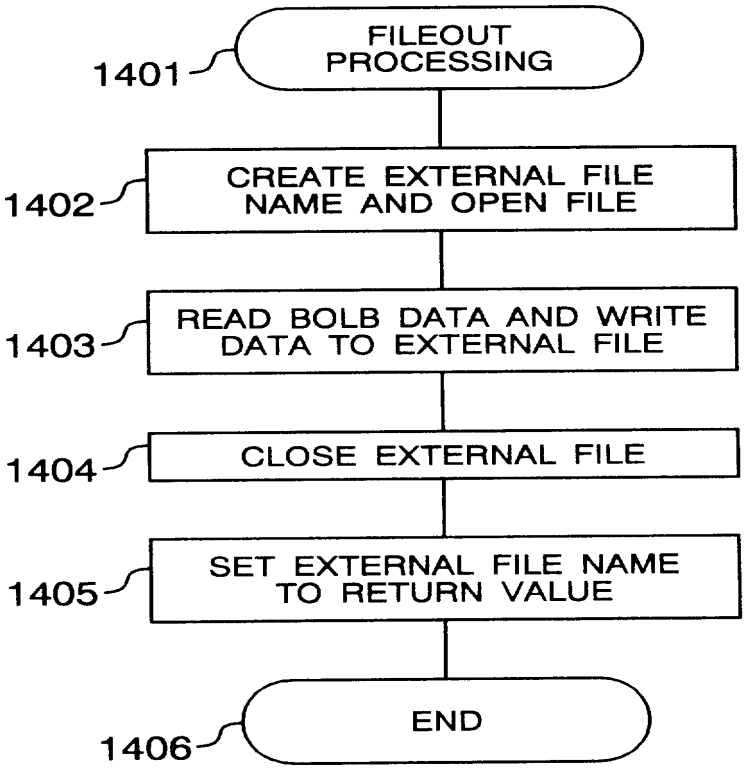

1401
FILEOUT PROCESSING
1402
CREATE EXTERNAL FILE NAME AND OPEN FILE
1403
READ BOLB DATA AND WRITE DATA TO EXTERNAL FILE
1404
CLOSE EXTERNAL FILE
1405
SET EXTERNAL FILE NAME TO RETURN VALUE
1406
END

FIG. 15

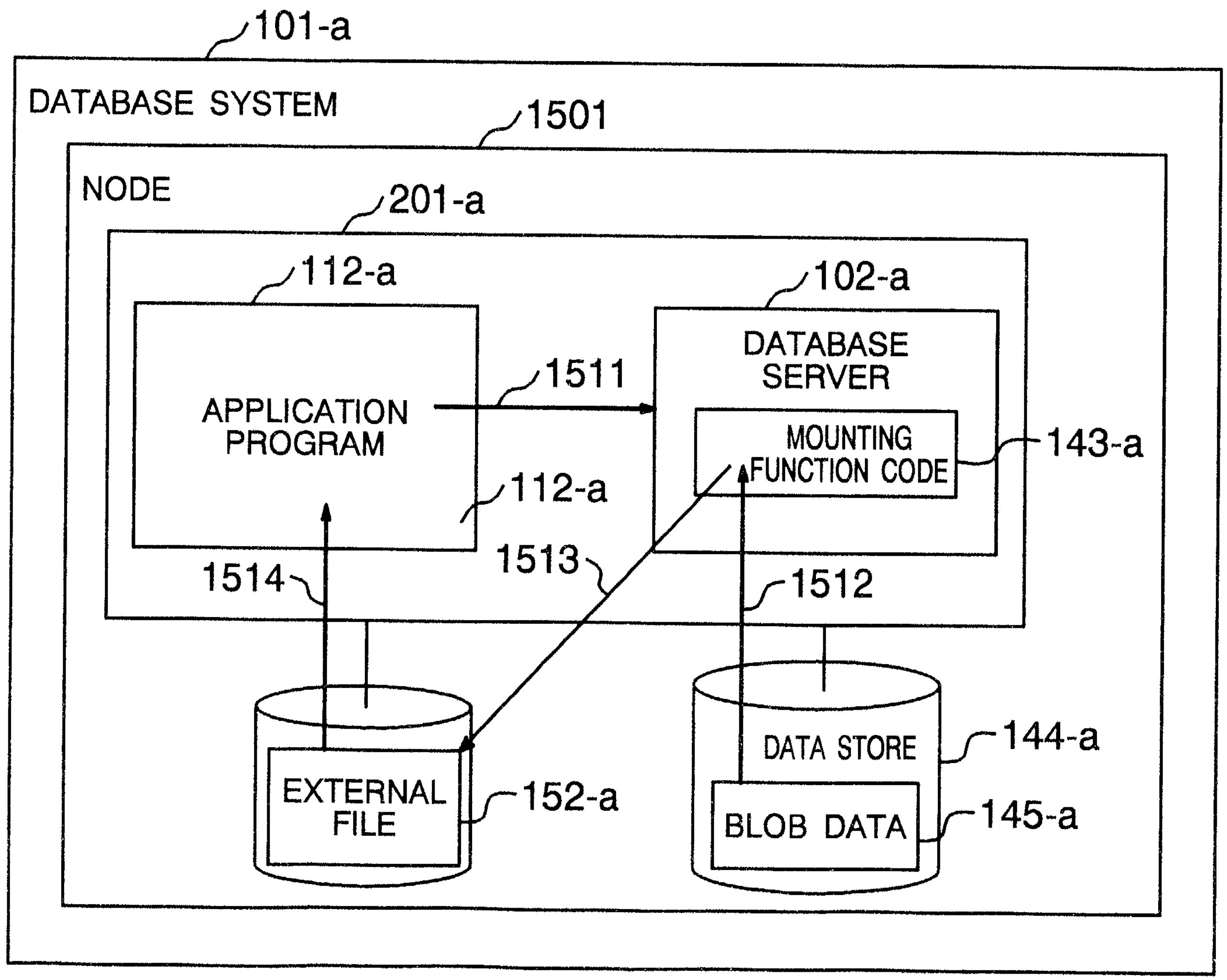
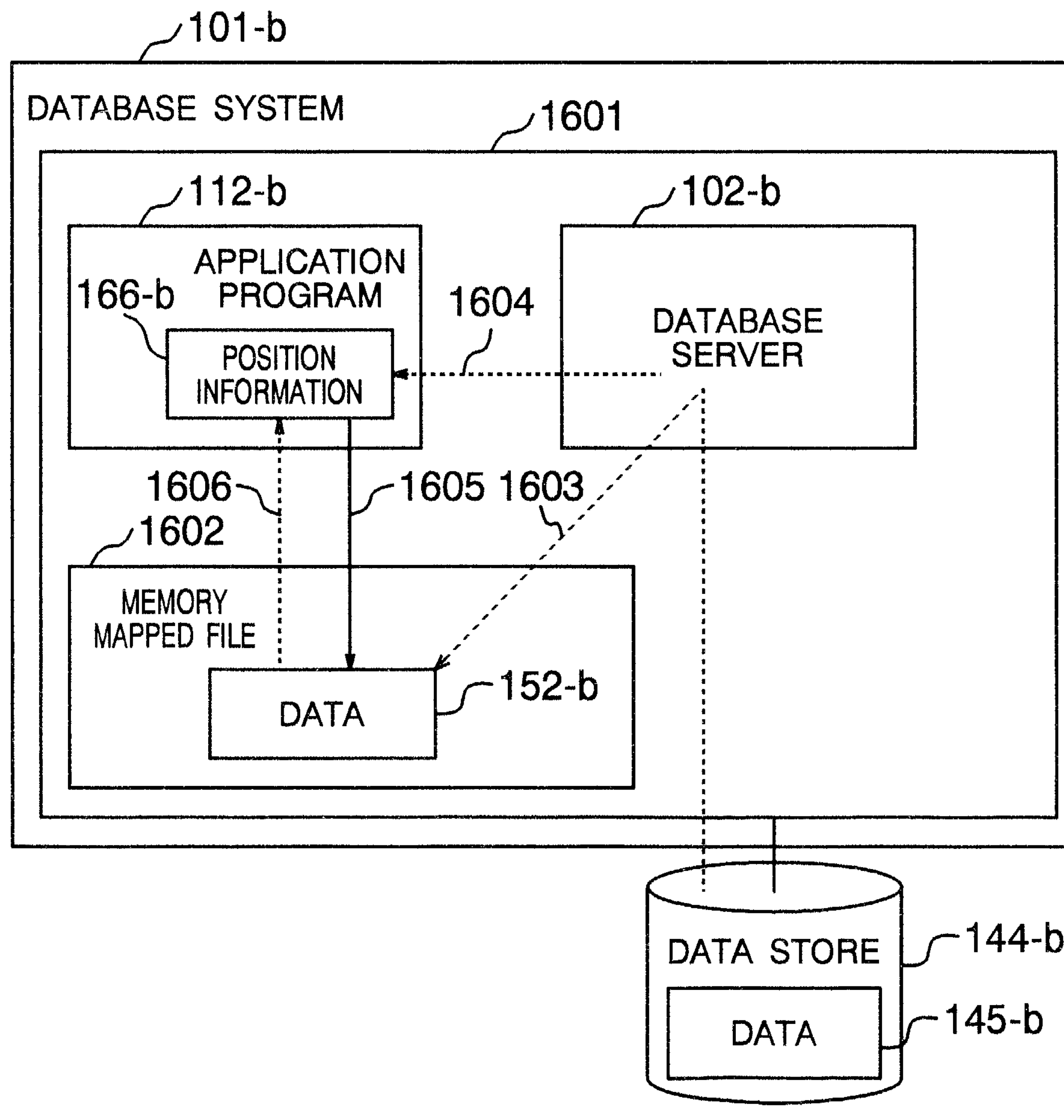

101-a
DATABASE SYSTEM
1501
NODE
201-a
112-a
APPLICATION
PROGRAM
112-a
102-a
DATABASE
SERVER
MOUNTING
FUNCTION CODE
143-a
1511
1513
1514
1512
EXTERNAL
FILE
152-a
DATA STORE
144-a
BLOB DATA
145-a

FIG. 16


101-b
DATABASE SYSTEM
1601
112-b
APPLICATION PROGRAM
166-b
POSITION INFORMATION
1604
102-b
DATABASE SERVER
1606
1605
1603
1602
MEMORY MAPPED FILE
DATA
152-b
DATA STORE
144-b
DATA
145-b

E4584-01
(*)

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name, I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

"METHOD AND SYSTEM FOR MANAGING DATABASE HAVING A CAPABILITY OF PASSING DATA, AND MEDIUM RELEVANT THERETO",

the specification of which (check one)

[X] is attached hereto.

[ ] was filed on _____
as Application Serial No. _____
and was amended on _____ (if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
10-147262 (Number)	Japan (Country)	28 May, 1998 (Day/Month/Year Filed)	[X] Yes	[ ] No
(Number)	(Country)	(Day/Month/Year Filed)	[ ] Yes	[ ] No
(Number)	(Country)	(Day/Month/Year Filed)	[ ] Yes	[ ] No
(Number)	(Country)	(Day/Month/Year Filed)	[ ] Yes	[ ] No
(Number)	(Country)	(Day/Month/Year Filed)	[ ] Yes	[ ] No
(Number)	(Country)	(Day/Month/Year Filed)	[ ] Yes	[ ] No

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Status: patented, pending, abandoned)
(Application Serial No.)	(Filing Date)	(Status: patented, pending, abandoned)
(Application Serial No.)	(Filing Date)	(Status: patented, pending, abandoned)
(Application Serial No.)	(Filing Date)	(Status: patented, pending, abandoned)

(Continued on Page 2)

E4584-01(*)

I hereby appoint as principal attorneys; Donald R. Antonelli, Reg. No. 20,296; David T. Terry, Reg. No. 20,178; Melvin Kraus, Reg. No. 22,466; Stanley A. Wal, Reg. No. 26,432; William I. Solomon, Reg. No. 28,565; Gregory E. Montone, Reg. No. 28,141; Ronald J. Shore, Reg. No. 28,577; Donald E. Stout, Reg. No. 26,422; Alan E. Schiavelli, Reg. No. 32,087; James N. Dresser, Reg. No. 22,973 and Carl I. Brundidge, Reg. No. 29,621 to prosecute and transact all business connected with this application and any related United States application and international applications. Please direct all communications to the following address:

Antonelli, Terry, Stout & Kraus
Suite 1800
1300 North Seventeenth Street
Arlington, Virginia 22209
Telephone: (703) 312-6600
Fax: (703) 312-6666

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United State Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

(Full Name) (Signature)

Date May 13, 1999 Inventor Susumu KOBAYASHI Susumu Kobayashi

Residence Kawasaki-shi, Japan Citizenship Japan

Post Office Address 810-B101, Ogura, Saiwai-ku, Kawasaki-shi, Japan.

Date May 13, 1999 Inventor Youichi YAMAMOTO Youichi Yamamoto

Residence Machida-shi, Japan Citizenship Japan

Post Office Address 1426-211, Minamioya, Machida-shi, Japan.

Date May 13, 1999 Inventor Yoshito KAMEGI Yoshito Kamegi

Residence Yokohama-shi, Japan Citizenship Japan

Post Office Address 2-14, Serigaya-4-chome, Konan-ku, Yokohama-shi, Japan.

Date _____ Inventor _____

Residence _____ Citizenship _____

Post Office Address _____

Date _____ Inventor _____

Residence _____ Citizenship _____

Post Office Address _____

Date _____ Inventor _____

Residence _____ Citizenship _____

Post Office Address _____

Date _____ Inventor _____

Residence _____ Citizenship _____

Post Office Address _____

Date _____ Inventor _____

Residence _____ Citizenship _____

Post Office Address _____

Date _____ Inventor _____

Residence _____ Citizenship _____

Post Office Address _____

Date _____ Inventor _____

Residence _____ Citizenship _____

Post Office Address _____